**Q1 .Explain term filehandling.Write sequence of operations along with python syntax to open a file in read mode.**

**ANS** Till now, we were taking the input from the console and writing it back to the console to interact with the user.

Sometimes, it is not enough to only display the data on the console. The data to be displayed may be very large, and only a limited amount of data can be displayed on the console, and since the memory is volatile, it is impossible to recover the programmatically generated data again and again.

However, if we need to do so, we may store it onto the local file system which is volatile and can be accessed every time. Here, comes the need of file handling.

**opening a file:**

Python provides the open() function which accepts two arguments, file name and access mode in which the file is accessed. The function returns a file object which can be used to perform various operations like reading, writing, etc.

The syntax to use the open() function is given below.

1. file object = open(<file-name>, <access-mode>, <buffering>)

The files can be accessed using various modes like read, write, or append. The following are the details about the access mode to open a file.

|  |  |  |
| --- | --- | --- |
| **SN** | **Access mode** | **Description** |
| 1 | r | It opens the file to read-only. The file pointer exists at the beginning. The file is by default open in this mode if no access mode is passed. |
| 2 | rb | It opens the file to read only in binary format. The file pointer exists at the beginning of the file. |
| 3 | r+ | It opens the file to read and write both. The file pointer exists at the beginning of the file. |
| 4 | rb+ | It opens the file to read and write both in binary format. The file pointer exists at the beginning of the file. |
| 5 | w | It opens the file to write only. It overwrites the file if previously exists or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file. |
| 6 | wb | It opens the file to write only in binary format. It overwrites the file if it exists previously or creates a new one if no file exists with the same name. The file pointer exists at the beginning of the file. |
| 7 | w+ | It opens the file to write and read both. It is different from r+ in the sense that it overwrites the previous file if one exists whereas r+ doesn't overwrite the previously written file. It creates a new file if no file exists. The file pointer exists at the beginning of the file. |
| 8 | wb+ | It opens the file to write and read both in binary format. The file pointer exists at the beginning of the file. |
| 9 | a | It opens the file in the append mode. The file pointer exists at the end of the previously written file if exists any. It creates a new file if no file exists with the same name. |
| 10 | ab | It opens the file in the append mode in binary format. The pointer exists at the end of the previously written file. It creates a new file in binary format if no file exists with the same name. |
| 11 | a+ | It opens a file to append and read both. The file pointer remains at the end of the file if a file exists. It creates a new file if no file exists with the same name. |
| 12 | ab+ | It opens a file to append and read both in binary format. The file pointer remains at the end of the file. |

Let's look at the simple example to open a file named "file.txt" (stored in the same directory) in read mode and printing its content on the console.

Example

1. fileptr = open("file.txt","r")
3. **if** fileptr:
4. **print**("file is opened successfully")

**Output:**

file is opened successfully

Reading the file

To read a file using the python script, the python provides us the read() method. The read() method reads a string from the file. It can read the data in the text as well as binary format.

The syntax of the read() method is given below.

1. fileobj.read(<count>)

Here, the count is the number of bytes to be read from the file starting from the beginning of the file. If the count is not specified, then it may read the content of the file until the end.

Consider the following example.

Example

fileptr = open(“file.txt”,”r”);

content = fileptr.read(9);

**print**(type(content))

**print**(content)

fileptr.close()

**Output:**

Hi, I am

Read Lines of the file

Python facilitates us to read the file line by line by using a function readline(). The readline() method reads the lines of the file from the beginning, i.e., if we use the readline() method two times, then we can get the first two lines of the file.

Consider the following example which contains a function readline() that reads the first line of our file **"file.txt"** containing three lines.

.

fileptr = open("file.txt","r”)

content = fileptr.readline();

**print**(type(content))

**print**(content)

fileptr.close()

**Output:**

Hi, I am the file and being used as

Looping Through File

By looping through the lines of the file, we can read the whole file.

Example

fileptr = open("file.txt","r");

**for** i **in** fileptr:

**print**(i)

**Output:**

Hi, I am the file and being used as

an example to read a

file in python.